Computer Communications 33 (2010) 1809–1822

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

A GRASP-based network re-optimization strategy for improving RWA in multi-constrained optical transport infrastructures

Francesco Palmieri [a,*], Ugo Fiore [a], Sergio Ricciardi [b]

[a] Università degli Studi di Napoli Federico II, CSI, Complesso Universitario Monte S. Angelo, Via Cinthia, 80126 Napoli, Italy
[b] Universitat Politècnica de Catalunya (UPC), Departament d'Arquitectura de Computadors (DAC), Jordi Girona 3, 08034 Barcelona, Catalunya, Spain

ARTICLE INFO

Article history:
Received 14 December 2009
Received in revised form 25 April 2010
Accepted 7 May 2010
Available online 13 May 2010

Keywords:
Network Re-optimization
GRASP meta-heuristic
Wavelength-routing
RWA
Grooming

ABSTRACT

In wavelength-routed optical networks, end-to-end connection demands are dynamically routed according to the current network status. Naïve path selection schemes, the wavelength continuity constraint and the limited or inaccurate information available can cause the virtual topology resulting from the currently allocated lightpaths to become sub-optimal. We propose an efficient re-optimization technique based on a GRASP meta-heuristic. Our work is focused on a hybrid online–offline scenario: connections are ordinarily routed dynamically using one of the available algorithms for online routing, but occasionally, when reorganization of the current virtual topology is desirable, existing paths are re-routed in order to improve load balancing and hence the ability to efficiently accept further connections. Because global changes of the logical topology and/or routing scheme can be disruptive for the provided connection services, we used iterative stepwise approaches based on a sequence of small actions (i.e., single connection re-routing and on local search from a given configuration). Simulation results demonstrate that several network performance metrics – including connection blocking ratios and bandwidth gains – are significantly improved by such approach. In particular, we achieved to accept more connection requests in our re-optimized networks with respect to the same networks without re-optimization, thus lowering the blocking ratio. Besides, in all tests we measured a notable gain in the number of freed bandwidth OC-units thanks to our re-optimization approach.

1. Introduction

While being attractive for their transparent and cost-efficient operation, all-optical networks need complex routing practices and accurate engineering of Wavelength Division Multiplexed (WDM) paths, to match the constraints of the underlying photonic technology with the requirements of the dynamic traffic flows that should be transported. Dynamic routing and wavelength assignment schemes commonly used within these infrastructures tend to lead to network inefficiencies due to the limited or inaccurate information available for online routing [1], to the simple path selection algorithms often used and to the wavelength continuity constraint [2]. Precisely, the paths for the arriving connection requests are calculated starting from the current network state, including all the already routed connections. As the network and traffic evolve, such routing solutions may become sub-optimal [2]. The evolution process may also lead to changes in network topology due to the addition/deletion of new links and/or changes resulting from customers varying demands for different services. Some connection requests may be rejected due to lack of capacity, while a more efficient routing scheme would have allowed successful routing and path set-up. Furthermore, dynamic online routing practices typically tend to unbalance resource usage over time, causing severe congestion on some "critical" links that are most likely needed for satisfying future traffic demands [3]. When this happens, routing optimality may be restored only by periodic offline re-optimization that re-routes some of the existing connections over alternative paths, recovering the stranded capacity and re-balancing the load on the links [2]. Nevertheless, there is a cost directly associated with re-optimization, both in terms of computational complexity and of disruption of active connections [4]. Thus, the re-optimization activity has to be prudently planned to maximize the recovered stranded capacity and an integrated approach for periodic offline re-optimization of optical networks with sub-wavelength traffic is desirable. The focus of this work has been the best balancing of the reconfiguration cost, in terms of both computational complexity and disturbance to the network, within the context of a flexible and effective RWA/grooming solution. To allow a joint consideration of routing and wavelength assignment (RWA) with grooming and reconfiguration/optimization costs, we modeled the problem as a multi-objective optimization problem and solved it heuristically through a greedy randomized adaptive

* Corresponding author. Tel.: +39 081676632; fax: +39 081676628.
E-mail addresses: francesco.palmieri@unina.it (F. Palmieri), ugo.fiore@unina.it (U. Fiore), sergio.ricciardi@ac.upc.edu (S. Ricciardi).

0140-3664/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.comcom.2010.05.003

search procedure (GRASP) [5,6] in conjunction with a path-relinking [7,8] solution refinement procedure. In our approach, the implementation of the GRASP-based re-optimization works at the control-plane layer on each involved network element and includes several novel greedy construction and local search strategies, and a new simplified form of path-relinking. Overall, the heuristic approach is streamlined through the incorporation of advanced network flow re-optimization techniques and is based on a totally flexible network model, supporting heterogeneous WDM equipment, in which the number and type of lambdas can be independently specified for each link. We evaluate the effectiveness of our approach by simulating the proposed re-optimization schema and measuring the freed bandwidth and the percentage of the connections that the network is able to route after the re-optimization process. Results indicate that this implementation may lead to significant improvements of the network in comparison with the existing dynamic RWA solution with an acceptable performance impact due to offline re-optimization. An especially appealing characteristic of this GRASP-based approach–that makes it particularly suitable for the re-optimization of large networks and preferable to other heuristics–is its straightforward implementation. A limited number of parameters need to be assigned and tuned, and consequently development can easily focus on implementing efficient data structures to speed GRASP iterations up. Finally, the GRASP solution-search strategy can be trivially implemented in parallel between the available network nodes. Each processor node has to be initialized with its own instance data and an independent sequence of random numbers needed by the GRASP procedure. All the GRASP iterations are then handled in parallel by using only a single shared global variable, required to store the best solution found over all processors, thus greatly reducing computational complexity and signaling overhead.

2. Background

This section briefly introduces some of the concepts that will be useful to better explain the proposed integrated dynamic RWA/re-optimization paradigm, by presenting the underlying architectural scenario, the basic building blocks, assumptions and modeling details together with the theory behind it.

2.1. Wavelength-routed optical networks

With WDM, a single optical fiber is shared by a number of independent wavelengths (channels), each of which may transparently carry signals in different formats and bit-rates, for example STM-16 and 10G-Ethernet. Over the physical topology composed of optical cross-connection (OXC) devices connected by fiber links, a quasi-static *virtual topology*is superimposed by interconnecting pairs of edge nodes with *lightpaths*, all-optical channels that are never converted into an electrical signal at intermediate nodes across the optical backbone. Edge nodes transform the optical signal in electrical form and route it on client subnetworks. More sophisticated, and costly, OXCs, besides switching specific wavelengths between ports, can also convert input wavelengths into different output wavelengths. Ordinarily, therefore, a lightpath uses the same wavelength on all the links along its route.

2.2. The routing and wavelength assignment problem

Every lightpath must be routed on the physical topology and assigned a wavelength: this process is called routing and wavelength assignment (RWA). In general, the RWA problem is characterized by two constraints specific of an optical network:

- the wavelength continuity constraint, i.e. a lightpath must use the same wavelength on all the links along its route;
- the wavelength clash constraint, i.e. two or more lightpaths using the same fiber link must be allocated distinct wavelengths.

The wavelength continuity constraint may be relaxed if OXCs are equipped with wavelength converters [13]. Different levels of wavelength conversion capability (full or limited) are possible, depending on the number of converter-equipped OXCs and to the number of wavelengths that can be converted in each node. Note that when using full wavelength conversion on each network node, the RWA problem reduces to the classical routing problem in a circuit-switched network. This work is quite general in that we make no assumption on the availability of wavelength converters.

With *static* traffic [9], the entire set of connection requests is known in advance, and so the problem is reduced to setting up permanent lightpaths while minimizing the number of wavelengths or the number of fibers. The RWA problem for static traffic can be formulated as a mixed-integer linear program [10], which is NP-complete [11]. The two sub-problems of routing and wavelength assignment can also be separately faced. A review of these approaches is given in [9]. Here, *Incremental* connection requests arrive in sequence and the lightpaths established to handle these requests remain in the network indefinitely. On the other hand, for *dynamic* traffic, a lightpath is set-up to satisfy each request as it arrives, and such lightpath is released after a finite amount of time (connection lifetime). The objective in both the incremental and dynamic traffic models is to minimize the blocking/rejection probability of each connection (also known as blocking factor), or equivalently maximize the number of connections that are established in the network at any time. The dynamic case is more complex and usually several properly crafted heuristics are used to solve the routing and wavelength assignment sub-problems separately [12]. Here, we have chosen to deal with incremental traffic, as it is a simplified environment in which the effects or the proposed strategy are less dependent on the statistical distribution of connection requests and can therefore be better evaluated.

2.3. Integrating RWA with grooming: from the overlay model to a unified control-plane architecture

Typically, the traffic demand is partitioned into multiple parallel requests (between the same node pairs) with different bandwidth requirements, varying from tens or hundreds of Mbps (e.g. STM-1 or Fast Ethernet) up to the full-wavelength capacity (e.g. 10 Gigabit Ethernet). At the network edge, end-to-end connection requests, sharing the same traffic flow characteristics in terms of termination nodes and Quality of Service (QoS) requirements and involving capacities significantly lower than those of the underlying wavelength channels, can be efficiently multiplexed, or "groomed," onto the same wavelength/lightpath channel. A typical control-plane paradigm for traffic grooming operates on a two-layer multiple model, i.e. an underlying pure optical wavelength-routed network and an independent "opto-electronic" time-division multiplexed layer built over it. At the optical layer, wavelength routing traditionally sets up an almost static logical topology that is then used at the IP layer for routing, with lightpaths handled as single IP hops. By integrated RWA we instead mean a combined wavelength routing and grooming optimization paradigm, taking into account the whole topology and resource usage information at both layers. We assume that appropriate protocols exist for the unified control-plane, accurately disseminating correct and up-to-date information about the network state to each node, as well as taking care of resource reservation, allocation,

and release. Reconfiguration of virtual topology may be carried out for two reasons:

- to re-optimize the virtual topology under a changed traffic pattern or even a different cost metric;
- to create a new topology capable of supporting the current traffic pattern, without using failed or out-of-service network components.

In this work, traffic grooming is accounted by considering each flow as reallocatable. We focus on reconfiguration for re-optimization, when the traffic pattern changes or some part of the network becomes congested. Some of the most relevant issues involved in optical network re-optimization are discussed below.

2.4. Network re-optimization

Sophisticated routing algorithms can keep achieve remarkably low connection reject rates. However, these algorithms do not scale well with the growth in network size. On the other hand, many of the simple and scalable path selection algorithms may cause routing inefficiencies, leading to "stranded" capacity [14]. Whatever the RWA algorithm, the resources dedicated to serve each new connection request are selected according to the current network state, which is, in turn, the result of routing the existing connections. Keeping the network load balanced may lead to effective algorithms achieving good results in terms of blocking probability. This is necessarily the result of some estimation on the distributions of forthcoming requests. That estimation may presume that future requests will adhere to a uniform or Poisson distribution, or that they will repeat the pattern delineated from the currently provisioned demands. However, as the network and traffic evolve, the actual distribution of requests and their sequence of appearance may substantially drift from the estimates, and the network load distribution may become unbalanced. Network re-optimization is usually needed to increase the network utilization and can be performed by re-scheduling the already available connections requests in two ways: either by changing the associated paths only or by changing both the paths and the starting times. The latter solution is not very desirable as it implies re-negotiating the connection set-up times with users, and for this reason we will not consider it in this work. The idea of re-optimization is not new in telecommunications: carriers routinely use reconfiguration to better manage their network and increase utilization, which in turn allows them to defer investments on new infrastructure. Reconfiguration can also be used to provide better service performance, for example, by re-routing services over shortest paths if such paths become available.

2.5. GRASP

GRASP, which first appeared in [5] and was later formalized in [6], is an iterative two-phase meta-heuristic. A meta-heuristic may be defined as an iterative master process that guides and adjusts the operation of subordinate heuristics in order to produce high-quality solutions. When exploring the solution space, sometimes one may get stuck into local optima, i.e. solutions that are good locally but not globally. Meta-heuristics strive to escape such local optima by different strategies: occasionally accept worse solutions, as in simulated annealing [15] or tabu search [8]; combine existing solutions through mutation and crossover following the idea of genetic algorithms [16]; generate new solutions, as in GRASP. Greedy choices are performed and measured by means of an immediate or greedy gain possibly leading to sub-optimal solutions. For overcoming this myopic behavior, a heuristic measure can be introduced to evaluate this gain. At each iteration, the first phase produces a solution through the use of a greedy randomized adaptive construction scheme. In the second phase, local search is applied to this solution, in order to obtain a local optimum in its neighborhood. GRASP meta-heuristic may be customized to solve any problem for which simple construction and local search algorithms are available. Enhanced versions of the basic GRASP meta-heuristic have been applied to a wide range of combinatorial optimization problems [17]. Several new components and techniques have extended the original GRASP scheme (reactive GRASP, parameter variations, bias functions, memory and learning, improved local search, path-relinking, hybrids). These components are presented and discussed in [17]. In particular, path relinking was first introduced as a tool to compound intensification and diversification strategies in the context of tabu search [7,18]. The strategy is formulated on the principles of evolutionary approaches but unlike conventional evolutionary techniques (e.g., genetic algorithms), it does not employ randomization to generate new solutions. Instead, it constructs them through a methodical exploration of trajectories that connect previously generated high-quality solutions. An in-depth description of path-relinking can be found in [19,20]. The first application of GRASP and path-relinking was undertaken by Laguna and Marti [21]. Since then, a few other applications have appeared that combine the two methodologies. Some applications use path-relinking as an intensification strategy within the GRASP procedure [22]; others apply path-relinking as a post-optimization step after the execution of GRASP [23]. Some authors considered both utilizations of the path-relinking strategy [24,25]. According to the survey work on GRASP by Resende and Ribeiro [17], path relinking is more effective when used as an intensification phase. In our implementation, we have chosen to use it at the end of each GRASP iteration in order to intensify the search around local optima.

2.6. Related work

Lightpath re-optimization techniques have been discussed in several works available in literature. The problem of re-routing existing lightpaths in a dynamic routing scenario was addressed in [26,2] by invoking the re-optimization step only when new requests are unable to find a feasible path and it becomes absolutely necessary to re-route some of the existing paths to free up capacity from the most crowded links. Alternatively, [4] models the effect of the reconfiguration phase in terms of packet loss and bases its reconfiguration policy on this penalty criterion. Some different approaches, such as [27,28] reconfigure the underlying virtual topology of the optical network, respectively according to an ILP optimization and a stepwise branch exchange process, to adapt it to changing traffic patterns. Other authors have proposed solutions suitable for static traffic demand and heuristics for long-term on-demand traffic flows [29–32]. In [31] the authors study the problem of re-optimizing lightpaths in resilient mesh optical networks, where connection requests are routed using a pair of disjoint primary and backup paths. In their re-optimization scheme, all paths are re-routed, regardless of their being primary or backup. They also considered the effects of re-routing only the backup paths. An approach to combine dynamic online routing in a connection-oriented network with an offline optimization module, which constantly rebalances the load in the network whenever a certain imbalance threshold is exceeded, has been examined in [32]. In this scenario, the network operator determines a re-balancing benefit indicating the amount of traffic that could additionally be routed if the current traffic were to be redistributed, by computing the gain in "network efficiency" that a potential re-optimization would yield. If a threshold is exceeded, i.e. the benefits of re-optimizing the network exceed the incurred costs of the flow re-routing, then re-optimization is performed. More recently, other formulations of

the rearrangement problem have been proposed, differing in the optimization objectives. Notably, in [33], the number of rejected new demands and re-routed lightpaths is minimized through the Lagrangean Relaxation and Subgradient Method, while Din [34] investigated the use of a genetic algorithm and of simulated annealing with the objective of minimizing the average weighted propagation delay.

3. Integrated re-optimization scheme for wavelength-routed networks

We propose a novel dynamic RWA strategy specifically conceived to allow lightwave networks to carry more traffic without adding capacity, through a two-stage scheme based on hybrid online routing and offline re-optimization. Online dynamic routing is used in the first phase: connection requests arriving to the edge nodes over time are immediately routed by using a quick RWA scheme such as min-hop or shortest/minimum-cost path with dynamic weights based on wavelength available capacity. If there are enough resources to accommodate it, the required connection is routed on an already existing lightpath with available capacity and adequate QoS characteristics, or a new lightpath is properly set-up; otherwise the request is rejected. Over time, requests for teardown of existing connections may also arrive, causing the release of the involved resources. However, at a certain time, the residual capacity between certain critical ingress–egress pairs may be insufficient to accommodate new requests, but a different allocation of connection routes would easily permit it. When this happens, the blocking factor may increase indefinitely so that the network seems to be completely saturated even if there are still a lot of available resources. Thus, we continuously monitor the blocking factor, that can be viewed as a good approximated measure of routing efficiency, and when it exceeds a specific threshold β, we invoke re-optimization to restore routing optimality by re-routing some of the already established connections. Alternatively, we can also trigger the procedure when a specific number of connection requests has been received or served starting from the last occurrence of the re-optimization process. In both cases we address the network re-optimization issue as a periodic maintenance measure, activated when a tunable utilization threshold is exceeded, aiming at continuously keeping as much free network resources as possible with minimum total disruptions to the ongoing service. The objective of the second phase of our approach is then to eliminate the blocking or unbalancing generated during the previous quick-and-dirty connection set-up phase. Managing the lightwave network during the reconfiguration phase is a very complex issue, as re-optimization involves path reorganization, which may originate disruption in some critical services carried over the network, and therefore must be implemented carefully. Hence, to keep re-optimization as efficient as possible, all the connection requests arriving during the re-optimization process are queued and served only after its completion. Furthermore, the chosen re-optimization strategy must be conceived in order to combine maximum gains in recovering stranded capacity with minimal impacts on the overall network performance. Re-optimization must support the ability to provide guaranteed fault-tolerance to resilient connections. Finally, in order to preserve the packet arrival sequencing, the re-optimization strategy should not require traffic flow splitting on multiple paths. The sequence of operations through which the virtual topology is reconfigured, and the number of connections/lightpaths affected by such activity, can have a substantial impact on both the performance and capacity that is needed in the process and on the optimality of the obtained solution. The corresponding minimization problem, known as the Reconfiguration Sequencing Problem is indeed NP-hard [35]. Thus a re-optimization solution that re-routes all the connections of the existing virtual topology (without disruption) while keeping the network well balanced, by redistributing load thus freeing sufficient available capacity between all the ingress–egress node pairs, has to be found through the use of some heuristic technique that must ensure an acceptable run-time complexity. A key feature of such heuristic must be the ability of setting up the new re-provisioned paths one-by-one *before* re-routing traffic on them and only releasing the resources on the old paths *after* the new ones are totally established, according to the "make-before-break" principle. In other words, we do not explicitly perform re-routing on predefined backup paths or support specific post-optimization restoration strategies but we propose a new heuristic-based strategy, to be triggered on a maintenance basis, for finding approximate solutions to this problem, starting from a simple greedy approach and improving the quality of the re-optimization performance by using local search, through a combination of GRASP and path-relinking.

3.1. The network model

We denote the network by a graph $G = (V,E)$ where V is the set of nodes and E the set of links. We make no specific assumption on the number of wavelengths per fiber, number of fiber on each link and on the presence of wavelength conversion devices on the network. All these parameters are fully and independently configurable at the network topology definition time. Instead, we require that all the network nodes operate under a unique control-plane and share a common network view by relying on a common link-state protocol that is used to distribute resource usage information. Furthermore, we assume that every connection is bi-directional and consists in a specific set of traffic flows that cannot be split between multiple paths. Each connection can be routed on one or more (possibly chained) existing lightpaths between its source and destination nodes, with sufficient available capacity or on a new lightpath dynamically built on the network upon the existing optical links. Grooming decisions are taken according to adaptive strategy that dynamically tries to fulfill the algorithm's network resource utilization and connection serviceability objectives by determining if the request can be routed on one of the available lightpaths, by time-division multiplexing it together with other already established connections, or, if there are no available resources to satisfy the request, a new lightpath is needed on the optical transport infrastructure. A network with m edge nodes supports bi-directional connection demands only between $m(m-1)/2$ source-sink node pairs (u, v) where source and sink nodes $u, v \in V$ are edge routers. These source-sink pairs can be numbered from 1 to M and for each source-sink pair (u, v) there may be an amount $d(u, v)$ of end-to-end bandwidth demand already provisioned in the network, measured by the aggregate bandwidth of all the connection flows between the source and sink pair. To simplify our model each connection request is only characterized by a QoS commitment on bandwidth, although it can be routed basing the decision on other QoS metrics such as limited latency, error rate, etc. that can be incorporated into Service Level Agreements by converting them into a bandwidth requirement as shown in [36]. In addition we denote by $D(u, v)$ the total desired demand for the source sink pair (u, v). For each link $e \in E$ in the network rb_e and mb_e denote respectively its current residual and total capacity.

Let P be the number of connection requests at re-optimization time, $c_k = (u_k, v_k, b_k)$, $k = 1, \ldots, P$, the generic kth connection request, where $u_k, v_k \in V$ are respectively the origin and destination and b_k the bandwidth required, and p_k the path servicing the connection c_k. A feasible solution to our RWA re-optimization problem is then the set $X = \{X_1, \ldots, X_P\}$, where the generic element X_k is a pair (c_k, p_k) describing the routing choice associated to each connection. The

actual routing p_k of a connection c_k is determined by means of a shortest-path computation, with a cost function that only depends on the residual bandwidth on the links. Therefore, routing of a connection only depends on the network state at the moment the connection is considered for routing. Note that, if all the connection requests are serviced sequentially starting from an empty network, the order of arrival uniquely determines the solution.

3.2. Grasp-based re-optimization

In order to find good approximate solutions to the above multi-objective optimization problem, we propose a methodology based on the combined use of GRASP and path-relinking. When implementing a GRASP procedure, several different issues need to be addressed and tailored to the structural characteristics of the problem under study. First, an adaptive greedy function needs to be defined to guide the iterative construction phase, which builds the solution by adding one element at the time. The greedy function is adaptive in the sense that its value must be updated after the insertion of each new element in the partial solution under construction in order to reflect the choice made. Second, a restriction mechanism must be defined to build the restricted candidate list (RCL), that is the list from which to select the next element to be added to the solution. A probabilistic selection strategy (random component) must then be specified to select an element from the RCL. Besides, the essential constituents of the local search procedure (i.e. the neighborhood structure N, the search strategy and the objective function) must be defined. Finally, the objective function for the optimization problem must be defined. The objective function may be aimed at minimizing or maximizing some quantities in order to optimize the problem resolution. We use a minimizing function, i.e. a function whose values must be kept as low as possible while respecting the problem constraints. The whole GRASP procedure is algorithmically sketched in Fig. 1.

Where $f: \mathscr{F} \to R$ is the objective function of a specific problem $\mathscr{P}$, mapping the set $\mathscr{F}$ of feasible solutions to real values in R. The neighborhood structure $\mathscr{N}$ relates a solution X of the problem to a subset of solutions $\mathscr{N}(X) \in \mathscr{F}$. The procedure consists of *MaxIter* iterations (lines 2–8) in which a new solution is built (line 3), its neighborhood is explored (line 4) and the objective function is evaluated on it looking for an improvement of the current best solution (lines 5–7). The construction phase (line 3) tries to build a new solution X' choosing randomly an element from the RCL. The local search explore the neighborhood $N(X')$ of the construction phase solution X' looking for a local optimum X'' such that $f(X'') \leqslant f(X')$. At the end of each step we compare the value of the objective function$f$ evaluated on the solution X" with X^* which is the best solution found till that moment and we eventually keep the better one as the best solution found; if the algorithm has achieved a local optimum X^* such that $f(X^*) \leqslant f(X)$ for all $X \in N(X^*)$, the best solution is updated with the new value. Finally, the best solution X^* found in all iterations is returned as the overall GRASP solution. GRASP may be also viewed as a repetitive sampling technique in which each iteration produces a sample solution taken from an unknown distribution of admissible ones, whose mean and variance depend on the restrictive nature of the RCL. Given an effective greedy function, the mean solution value is expected to be good, but probably sub-optimal. That is, if the RCL is restricted to a single element only, then the same solution will be always produced on all the iterations. Clearly, in this case, the variance will be zero and the mean will exactly match with the value of the greedy solution. If we impose a less restrictive limit on solutions cardinality, i.e. more elements are allowed in the RCL, then many different solutions will be produced, with a larger variance. The size of the RCL controls, then, the tradeoff between the randomness and greediness of the solution. Hence, the value of the parameter α, which regulates the RCL size as explained in the section below, has to be chosen carefully. The lesser the role of greediness as compared to randomness, the worse should the optimality of the average solution be. However, the best solution found outperforms the average and very often is optimal.

```
procedure GRASP(X, MaxIter, α, 𝒩)
Input:  current network state X
        maximum number of GRASP iterations MaxIter
        RCL parameter α
        neighborhood function 𝒩
Output: new network state X*
1.  X* ← Ø
2.  for i = 1 to MaxIter do
3.      X' ← BuildSolution(X, α)
4.      X'' ← LocalSearch(X', 𝒩)
5.      if f(X'') ≤ f(X*) then
6.          X* ← X'
7.      end
8.  end
9.  return X*
10. end procedure GRASP
```

Fig. 1. A generic GRASP algorithm.

3.3. The construction phase

In the construction phase, connections are routed one at a time, thus building the solution. The pseudo-code of the Greedy (lines 2 and 4) randomized (line 5) adaptive (line 7) search procedure is illustrated in Fig. 2. First, we sort the connection requests in non-increasing greediness into a list L according to the greedy criterion (line 2); then we start building the solution adding one connection request at a time till the whole candidates are routed (lines 3–8). At each iteration, the list L is restricted into the RCL containing only the first k elements of L (line 4) and a new connection request is randomly selected from the RCL (line 5) and routed in the network (line 6). To drastically reduce the computation times, we combined the strategies commonly used by GRASP and heuristic-biased stochastic sampling [22]. At each iteration, the list is reordered taking into account the choice made at the previous step and the RCL is formed again (line 7). The greedy criterion consists in assigning a highest greediness to the un-routed origin–destination pairs whose source and destination nodes have the largest residual bandwidths on their incident arcs together with a high value of the bandwidth required for the connection. Such strategy allows the requests between nodes that have most residual capacity and that have higher bandwidth demands to be served first.

In detail, we start from an empty solution vector. The P connection requests $c = (u, v, b)$ are ordered according to the greedy adaptive criterion Γ. For each node v, let us denote by $\delta(v)$ the cut separating v from the rest of the graph, i.e. the set of all incident arcs to v

$$\delta(v) = \{u \in V | [u, v] \in E\} \tag{1}$$

where $[u, v]$ denotes an un-directed arc in the graph G and by

$$\gamma(v) = \sum_{a \in \delta(v)} rb_a \tag{2}$$

the sum of the residual capacities rb_a over all the arcs a incident to v. The ordering criterion for a connection $c = (u, v, b)$ will be based on the value:

$$\Gamma(c) = \gamma(u) + \gamma(v) + b \tag{3}$$

The bandwidth term b has the purpose of prioritizing demands that have higher bandwidth requirements and letting smaller ones to be served later as they are easier to be routed. Note that the criterion is adaptive: the sorted list L may be rearranged as a consequence of

procedure BuildSolution(X, α)
Input: current network state X
RCL parameter α
Output: new candidate solution X
1. $X \leftarrow \emptyset$
2. Order the candidate connection requests $c = (u, v, b)$ in the list L according to the greedy criterion $\Gamma(c) = \gamma(u) \dashv \gamma(v) \dashv b$
3. **while** $L \neq \emptyset$ **do**
4. $RCL \leftarrow$ MakeRCL(α)
5. $v \leftarrow$ RandomSelect(RCL)
6. $X \leftarrow X \cup \{v\}$
7. Reorder candidates reflecting the choice made
8. **end**
9. **return** X
10. **end procedure** BuildSolution

Fig. 2. The solution construction algorithm.

the successfully routing of the chosen connection request*c*. In fact, when the request*c* is eventually routed – using the same cost function and routing algorithm of the previous phase – the residual bandwidth will decrease by b on all the involved arcs and the values computed by the greedy function will change reflecting the *new* available bandwidth on each arc along the path. For each connection request that is chosen from the list and routed, the greedy value has to be recomputed only on the connections whose extremes are involved in the routes of the previous connection. The Restricted Candidate List *RCL* is then built, selecting only the first *k*elements in the ordered list L, where k is determined by the value of a tuning parameter $\alpha \in [0,1]$, according to the following formula:

$$k = (1 - \alpha) + \alpha \cdot |L| \tag{4}$$

where $|L|$ is the (whole) candidate list size. Among all the elements in the RCL, one is randomly chosen to become the next component of the solution being constructed. This process is iterated until the vector is complete. Note that, as α makes k vary proportionally in $\{1,\ldots,|L|\}$, it allows us to control the amount of greediness and randomness in the choice of the next element to be added in the solution under construction. In particular, when $\alpha = 0$, $k = 1$ corresponding to a completely greedy choice. On the other side, when $\alpha = 1$, $k = |L|$ so that the whole list is selected and the choice is totally random. The randomness factor allows widely different solutions to be constructed at each GRASP iteration, helping us to avoid being trapped into local maxima in the solution space. On the other side such selection strategy does not necessarily compromise the effectiveness of the adaptive greedy component of the method, as only the best-rated elements (on top of the list) can be chosen. It may happen that the construction phase fails, i.e., a state is reached where the current connection cannot be routed. In this case, the construction phase is restarted from scratch in the next iteration.

3.4. Performing local search on the solution space

Since the solutions generated by a GRASP process are not guaranteed to be locally optimal, it is almost always beneficial to attempt at improving each constructed solution by means of a local search in the solution space. A local search algorithm operates according to an iterative scheme, sequentially replacing the current solution with a better one found in the neighborhood of the current solution. The algorithm terminates when no better solution can be found in the neighborhood. A solution X is said to be locally optimal if in its neighborhood $\mathcal{N}(X)$ there are no solutions better than X. A significant limitation of local search is the risk of getting trapped into local optima. To circumvent this limitation, local search has to be driven by general-purpose heuristic strategies aiming at avoiding this phenomenon. The key success factors for such local search strategies are a good starting solution, the suitable choice of a neighborhood structure, and an efficient neighborhood search technique. It should be noted that each solution built in the previous phase might be viewed as a set of routes, one for each end-to-end connection request, where a single lightpath on the optical network must support one or more routes and a single route must use one or more lightpaths. The operation of constructing the solution neighborhood can be expressed as the construction of new "neighbor" network states resulting from the removal and re-routing of a single connection request at a time. For each neighbor, one connection c_i is selected, the resources allocated to c_i are released, and c_i is routed again, possibly along a different path (as the current network state is different from the one at which c_i was originally routed). This strategy ensures a low complexity for the neighborhood generation operation.

The network state is defined by the vector $\vec{S} = (rb_1, \ldots, rb_m)$ of the residual bandwidth on each of the m links. We can represent the route associated to a connection $c = (u, v, b)$ in the network state $\vec{S}$ by the *route* function:

$$route(c, \vec{S}) = (b_1^{(c)}, \ldots, b_m^{(c)}) \tag{5}$$

where $b_i^{(c)}$ is the bandwidth requested by the connection request c on the link i. Note that $b_i^{(c)}$ will be equal to the requested bandwidth b on the links along the route assigned to c and 0 on the other links. We then define a generic allocation function for a connection request c and a state $\vec{S}$,

$$allocate(c, \vec{S}) = \vec{S} - route(c, \vec{S}) \tag{6}$$

Let us consider a fixed order of arrival of the connection requests $c_1 \ldots, c_j$. The progressive routing of the succession of connections leads to a sequence of states. The initial state $\vec{S}_0$ corresponds to the starting condition in which every link is unused (empty network), so that:

$$\vec{S}_0 := (mb_1, \cdots, mb_m) \text{ where } mb_i \text{ is the maximum (initial) bandwidth for each link } i \tag{7}$$

and the generic state $\vec{S}_i$ is given by

$$\vec{S}_i := \vec{S}_{i-1} - route(c_i, \vec{S}_{i-1}) \tag{8}$$

or, equivalently, by

$$\vec{S}_i = \vec{S}_0 - \sum_{k=1}^{i} route(c_k, \vec{S}_{k-1}) \tag{9}$$

The *release* function is the complementary operation to the *allocate* function (6); let us first see what happens if we revert the last allocation in the sequence:

$$release(c_i, \vec{S}_i) = \vec{S}_{i-1} = \vec{S}_i + route(c_i, \vec{S}_{i-1}) \quad \text{by Eq.(8)} \tag{10}$$

In the general case,

$$release(c_i, \vec{S}_P) = \vec{S}_P + route(c_i, \vec{S}_{i-1}) \tag{11}$$

Therefore, we generate a new state with the following sequence:

$$\begin{aligned}\vec{S}_P &= allocate(c_i, release(c_i, \vec{S}_P)) \\ &= release(c_i, \vec{S}_P) - route(c_i, release(c_i, \vec{S}_P))\end{aligned} \tag{12}$$

We propose two local search procedures: breadth and depth local search. In order to build the solution neighborhood to be explored by breadth local search, we work as follows:

- for each connection request c_k, we start by removing the corresponding units of flow from each edge in its current route p_k;
- next, we calculate the resulting network status by determining the new edge weights. A tentative new shortest path route for the connection c is then computed by using the resulting weights.

Thus, the breadth local search process consists in releasing and reallocating every connection in the network and evaluating the objective function on the new network configurations obtained, keeping from time to time the best solution found. When every connection has been processed, the local search process ends and the best solution found is returned. In the depth local search strategy, whenever a better solution X_b is found in the neighborhood of the previous solution X, the local search process is repeated starting from X_b instead of X; the process is iterated for each connection request until all the candidates have been processed. The local search procedure pseudo-code is illustrated in Fig. 3. Each connection request $c = (u, v, b)$ (line 2) is extracted one at a time from the solution X (line 3) obtained by the construction phase and eventually re-routed in the network (line 4). Then, the objective function is evaluated on the new solution X' (line 5) and the best solution X^* is eventually updated with the new value (line 6). If the depth local search is chosen, the next iteration will start its local search from the new solution X'; otherwise, a breadth local search will be performed, starting from X.

3.5. The objective function

The ultimate objective of the offline re-optimization problem is to minimize the lightpath rejection or delayed creation (due to the duration of the re-optimization process) while balancing the load on the optical links (and hence maximizing network resources utilization). A good balancing can be achieved by minimizing the load on the most utilized fiber trunks. Of course, routing and wavelength assignments with minimum delays may not attain the best load balance within the network and, likewise, RWA algorithm realizing the best load balance may not minimize creation delays or connection rejection at all. We essentially aim at routing the connections in such a way that a desired point in the tradeoff curve between creation delays and network load balancing is achieved. Hence, the objective function we use to compare the solutions found in the previous phase should provide an extremely effective metric for evaluating the degree of network resource load balancing together with an acceptable run-time performance, to avoid, as much as possible, the excessive delay of queued requests that may arrive during the re-optimization phase. The chosen objective function value clearly determines how the virtual topology is best suited for the given traffic demand. When the traffic pattern changes the network state may not remain optimal and the virtual topology needs to be changed to reflect the objective function goals. This change requires reconfiguration of the network components (OXCs and routers) to establish the lightpaths present in the desired new virtual topology but absent in the current one. Similarly, the lightpaths that are not present in the new virtual topology must be torn down. Obviously, such reconfiguration has an operational cost that cannot be ignored. Thus, the best reconfiguration solution is a tradeoff between the improvement in the objective function value and the number of changes to the virtual topology needed to achieve that improvement. A natural choice for our objective function f is the variance of the load vs. capacity ratios for each link (with the minus sign accounting for the fact that we are trying to balance the load as evenly as possible):

$$f(\vec{S}) = Var\left(1 - \frac{rb_i}{mb_i}\right), \quad i \in \{1, \ldots, |E|\} \tag{13}$$

We can note that such choice reflects both effectiveness in describing load balancing and ease of computation so that it contributes to keep the re-optimization delay for pending connections as low as possible. The structure of the objective function is such that as the load balance of the network increases, its maximum utilization rate decreases, providing a useful strategy to achieve the QoS level defined by a desired maximum utilization rate. Finally, once a target solution has been chosen, the current allocation is transformed into the desired one by parallelly signaling (with the RSVP-TE) only the affected elements, so that minimal impact in terms of service disruption is achieved.

```
procedure LocalSearch(X, 𝒩, localSearchType)
Input:  the solution built by the construction phase X
        neighborhood function 𝒩
        local search type parameter localSearchType
Output: new network state X*
1.  X* ← Ø
2.  for i = 1 to |X| do
3.      X' ← X \ {c_i}
4.      X' ← X' ∪ {c_i}
5.      if f(X') ≤ f(X*) then
6.              X* ← X'
7.              if localSearchType = "depthSearch" then
8.                      X ← X'
9.              end
10.     end
11. end
12. return X*
13. end procedure LocalSearch
```

Fig. 3. The local search algorithm.

3.6. Path-relinking

Path-relinking may be viewed as an elite selection strategy aiming at adding in new solutions only high quality attributes, by privileging these attributes in the determination of other solution that best improve (or least deteriorate) the initial one. It works on a population of already good solutions by properly combining them to obtain new (better) ones. Such new solutions are then generated by exploring the trajectories connecting high-quality solutions and the name path relinking is used because the involved solutions are linked by a series of transformations, performed during the search process, relinking previous points in ways not already obtained in previous search history. Both a source and a guide solution have to be selected in order to generate these paths in the solution space. We can also start from a set of guiding solutions (multiple parents) generating combinations of elite solutions that link the points in the solution space in several ways. During the process of linking a solution X (initial reference point) to a solution Y (desired or guiding point), a path is constructed by the greedy selection of re-routing actions with respect to the evaluation of the objective

function. Simply stated, a transformation is selected if it locally maximizes the objective function value. The main objective of path-relinking is the incorporation of attributes belonging to the guiding solution (or solutions) while recording values of the objective function. The purpose of these actions is to obtain several improved solutions within the neighborhood of the already visited ones. The trajectory from X to Y is generated iteratively, by selecting the greedy X neighbor solution (we will call this solution by Z), from a set of all neighbors that decrease the distance from X to Y. This distance may be determined by calculating the number of differences between the solutions (X and Y). The procedure is restarted, making $X \leftarrow Z$, until the target solution Y is obtained. Path-relinking is applied to pairs of solutions (X_1,X_2), where X_1 is the locally optimal solution, obtained through local search, and X_2 is randomly chosen from a pool of at most *MaxElite* elite solutions found along the search process. Such pool is originally empty. Each locally optimal solution obtained during the local search can be considered as a candidate to be inserted into the above pool only if it is different (by at least one link utilization in one route) from every other solution already present in the pool. If the pool already contains *MaxElite* solutions and the candidate is better than the worst of them, then the former replaces the latter. Otherwise, if the pool is not full, the candidate solution is simply inserted into it. This procedure is iterated until no further change in the pool occurs. Such type of intensification can be done in a post-optimization phase (by using the final elite solutions pool), or periodically, during the optimization process (by using the current elite solutions set). When path-relinking is used in a post-optimization phase, the local search procedure is applied to each elite solution when no further occurs change in the elite set, since the solutions produced by path-relinking are not always local optima. All the local optima found during local search are candidates for insertion into the elite set. The entire post-optimization process is repeated until changes occur within the elite set. In detail, our path-relinking algorithm starts by computing the symmetric difference $d(X_1,X_2)$ between X_1 (the initial solution) and X_2 (the guiding solution), resulting in the set of re-routing actions, which should be applied to the first one to reach the other. Then, by starting from the initial solution, the best topology change still not performed is applied to the actual solution, until the guiding one is reached. The best solution found along this trajectory is also considered as a candidate for insertion in the elite pool. Since the neighborhood of the initial solution is more carefully explored than that of the guiding one, starting from the best of them gives to the algorithm a much better chance of investigating in more details the neighborhood of the most promising solution [22]. Path-relinking can be applied to a pure GRASP procedure in a straightforward manner, as it can be seen in the integrated GRASP/Path Relinking algorithm reported in Fig. 4.

First, the set of elite solutions $\mathscr{E}$ and the best solution X^* are initialized to the empty sets (lines 1–2) and $\mathscr{E}$ is built by including the solutions from the first *MaxElite* iterations (lines 9–10). After that a standard GRASP iteration produces a local optimal solution X'' (lines 3–5), the PathRelinking procedure is called (line 7). Then, a function UpdateElite is called (line 8) in which the elite pool is possibly updated. The solution returned from path-relinking is included in the elite pool if it is better than the best solution in $\mathscr{E}$ or if it better than the worst and is sufficiently different from all elite solutions [37]. Finally, the optimal solution is updated if necessary (lines 12–14).

procedure GRASPwithPathRlnk(X, *MaxIter*, α, $\mathcal{N}$)
Input: current network state X
maximum number of GRASP iterations *MaxIter*
RCL parameter α
neighborhood function $\mathcal{N}$
Output: new network state X^*
1. $X^* \leftarrow \emptyset$
2. $\mathscr{E} \leftarrow \emptyset$
3. **for** $i = 1$ **to** *MaxIter* **do**
4. $X' \leftarrow$ BuildSolution(X, α)
5. $X'' \leftarrow$ LocalSearch(X', $\mathcal{N}$)
6. **if** $|\mathscr{E}| =$ *MaxElite* **then**
7. $X'' \leftarrow$ PathRelinking(X'', $\mathscr{E}$)
8. UpdateElite(X'', $\mathscr{E}$)
9. **else**
10. $\mathscr{E} \leftarrow \mathscr{E} \cup \{X''\}$
11. **end**
12. **if** $f(X'') \leq f(X^*)$ **then**
13. $X^* \leftarrow X''$
14. **end**
15. **end**
16. **return** X^*
17. **end procedure** GRASPwithPathRlnk

Fig. 4. GRASP with path-relinking re-optimization algorithm.

4. Complexity analysis

Let's now examine the computational complexity of the above GRASP-based optimization framework. We consider a network with n nodes and up to λ_{max} wavelengths on each of the m fiber links in which P connection requests have already been routed on the existing lightpaths. First, we remark that the computational complexity associated to the re-routing of a single connection in the construction phase of each solution is given by the routing algorithm used, in our case the traditional Shortest Path First (SPF) algorithm ($O(m \cdot \lambda_{max} \cdot \log n)$ by using a priority queue with a Fibonacci heap in the implementation of the Dijkstra algorithm).

The computational complexity of the GRASP procedure can be calculated by summing, for each iteration, the complexity of all its component procedures (in the following indicated as $\langle Procedurename \rangle_C$) and is given by

$$\begin{aligned}\text{GraspProcedure}_C = O(&MaxIter \cdot (\text{BuildSolution}_C + \text{LocalSearch}_C \\ &+ \text{ObjectiveFunction}_C))\end{aligned} \tag{14}$$

In the *BuildSolution* procedure, the initial sorting of the list L costs $O(P \log P)$, the *while* cycle repeats P times its body that consists of *MakeRCL* and *RandomSelect* that are constant time operations $O(1)$, a SPF routing $O(m \cdot \lambda_{max} \cdot \log n)$ and a partial reordering that may be reduced to a simple optimized update with a worst case cost of $O(n \cdot \log n)$, because we know exactly what are the elements whose value may only decrease. So, the complexity of the *BuildSolution* procedure is given by

$$\begin{aligned}\text{BuildSolution}_C &= O(\text{Sorting}_C + P \cdot (\text{MakeRCL}_C \\ &\quad + \text{RandomSelect}_C + \text{Routing}_C + \text{Update}_C)) \\ &= O(P \cdot \log P + P \cdot (2k + m \cdot \lambda_{max} + n \cdot \log n \\ &\quad + n \cdot \log n)) \\ &= O(P \cdot \log P + P \cdot (m \cdot \lambda_{max} + n \cdot \log n))\end{aligned} \tag{15}$$

The *LocalSearch* procedure repeats P times its cycle that consists of a release operation, a route SPF algorithm $O(m \cdot \lambda_{max} + n \cdot \log n)$ and one evaluation of the objective function. The release operation has to free the bandwidth on all the links crossed by the connection, that may be $n - 1$ in the worst case for a network without cycle paths, so it costs $O(n)$. The objective function has to evaluate the bandwidths on the network's edge, thus it costs $O(m \cdot \lambda_{max})$ in the worst case. Thus, the computational cost of the *LocalSearch* procedure is

$$\begin{aligned}\text{LocalSearch}_C &= O(P \cdot (\text{Release}_C + \text{Route}_C \\ &\quad + \text{ObjectiveFunction}_C)) \\ &= O(P \cdot (n + m \cdot \lambda_{\max} + n \cdot \log n + m \cdot \lambda_{\max})) \\ &= O(P \cdot (m \cdot \lambda_{\max} + n \cdot \log n)) \end{aligned} \quad (16)$$

Consequently, the overall *GRASP* procedure costs:

$$\begin{aligned}\text{GraspProcedure}_C &= O(MaxIter \cdot (\text{BuildSolution}_C \\ &\quad + \text{LocalSearch}_C + \text{ObjectiveFunction}_C)) \\ &= O(MaxIter \cdot (P \cdot \log P + P \cdot (m \cdot \lambda_{\max} + n \\ &\quad \cdot \log n) + P \cdot (m \cdot \lambda_{\max} + n \cdot \log n) + m \\ &\quad \cdot \lambda_{\max})) \\ &= O(MaxIter \cdot (P \cdot \log P + 2P \cdot (m \cdot \lambda_{\max} + n \\ &\quad \cdot \log n) + m \cdot \lambda_{\max})) \\ &= O(MaxIter \cdot (P \cdot \log P + P \cdot m \cdot \lambda_{\max} + P \\ &\quad \cdot n \cdot \log n)) \end{aligned} \quad (17)$$

For typical topologies and traffic values (e.g. $n = 30$, $P = 3000$), $n \cdot \log n$ is the dominant factor with respect to $\log P$ as, even in the case in which P is much greater than n, the logarithm function will weight very little its argument while the dominant factor will be the multiplicative n (in the example, $30 \cdot \log 30 = 44.3$, $\log 3000 = 3.5$). Thus the worst case complexity of the whole optimization process may be simplified as

$$O(MaxIter \cdot P \cdot (m \cdot \lambda_{\max} + n \cdot \log n)). \quad (18)$$

As we have illustrated, the GRASP re-optimization is based on an iterative improvement process (represented essentially by the *MaxIter* factor) whose computational complexity may be high for large-scale RWA/grooming networks [38]. Parallel local search algorithms, when applicable, are an effective way to cope with this problem. According to an iteration decomposition principle, the search iterations can be partitioned into several threads and the main procedure run in each of them in parallel. In our GRASP-based approach, the *MaxIter* iterations may be easily distributed among the network nodes and run in parallel, thus cooperating to implement the integrated RWA mechanism within the network control-plane logic (following the so called *multiple-walk independent-thread* strategy, based on distributing the GRASP iterations over the available processors, that in our case are the switching nodes). Each processor works on an independent copy of the problem data, and has an independent seed to generate its own pseudorandom sequence number. Clearly, each processor must base its work on a different pseudorandom sequence, to avoid the same solutions to be found by each of them. A single global variable, whose value can be kept synchronized between all the processors through proper message passing, is required for storing the best solution found by all the participating processors. Here, the communication among the different processors running the GRASP iterations in parallel is limited to the random seed and to the current best solution found. One of the processors acts as the master, by generating the random seeds to be used on each processor, reading and distributing the problem data and the iterations, and finally collecting the best solution found by each computing node. Since all the iterations are completely independent and very little information is exchanged between the participants, linear speedups can be easily obtained provided that no major load imbalance problems occur. To further improve load balancing, the iterations may be uniformly distributed over the processors according to their demands. From extensive simulations, it has been observed that a typical value is *MaxIter* = 30, that is in line with the mean number of network nodes in MAN/WAN network. In general, if we assume that each processor core runs one search thread, the computational complexity of the parallel *GRASP* procedure is decreased to

$$O(P \cdot (m \cdot \lambda_{\max} + n \cdot \log n)) \quad (19)$$

The time evaluation tests we conducted have showed that it is an affordable time complexity for a single processor thus confirming the feasibility of such approach. Parallel implementations of GRASP may also be used in conjunction with path-relinking. In the multiple-walk independent-thread implementation described by Aiex et al. [24], each processor applies path-relinking to pairs of elite solutions stored in a local pool. The OSPF opaque Link-State Advertisement (LSA) mechanism can be easily used to transport synchronization information between the cooperating nodes.

5. Performance evaluation and considerations

In this section, we examine the performance of our new algorithm with an extensive simulation study, by working on several real network topologies, with and without the continuity constraint (i.e. wavelength converters supported or not). The simulation details together with the most interesting results and observations emerged from the experiments have been reported in the following paragraphs.

5.1. The simulation environment

In order to evaluate the performance of the proposed hybrid online/offline routing framework we realized a simple and very flexible ad hoc optical network simulation environment written in Java in order to take advantage of its extensibility, ease of modifiability, portability and strict math and type definitions [39]. To allow us to perform a simple comparative analysis the above environment supports discrete-event simulations in fiber/lambda switched networks for several well-known RWA algorithms, both Dijkstra based, such as Minimum Hop Algorithm (MHA) and Shortest-Widest Paths (SWP), or interference-based, such as Maximum Open Capacity Algorithm (MOCA). It supports flexible definition and modification of simulation parameters and configuration files to define complex simulation test cases, allowing the creation of new network topologies. Simulations have been performed on an HP® DL380 Dual Processor (Intel® Xeon® 2.5 GHz) server running FreeBSD® 4.11 operating system and Sun® Java® 1.4.2 Runtime Environment by using several optical network topologies modeled as un-directed graphs in which each link has a non-negative capacity. In all the experiments, we used an incremental traffic model in which connection requests, defined by a Poisson process, arrive with a rate of δ requests/s and are distributed on the available network node according to a random-generated or predefined traffic matrix. Thus, the session holding time has been set to be infinite to enhance the effect of connection's load on the network, that is, each connection lasts through the entire simulation, letting the network resources saturating more rapidly. This can be done since dynamic connection releases do not adversely affect both the performance and behavior of the whole RWA framework whose periodic re-optimization steps are managed offline. The above choice allowed us to make our tests under the worst-case conditions.

5.2. Results analysis

The results presented are taken from many simulation runs on several network topologies with various GRASP parameter values and an increasing number of connection requests varying from 0 to 1000. The GRASP parameters and bandwidth unit request values used in our simulations are reported in Table 1.

Table 1
Simulations performed and parameters used.

Parameters	NSFNET/GEANT2
Number of connections	Varying from 0 to 1000 (step 100) 50% before and 50% after re-optimization
Random generated bandwidths (OC-unit)	{1,3,12} with different distribution probability
MaxIter	30
α	{0.2, 0.5, 0.8}
Local search	Breadth and depth local searches
Number of simulations	20 simulations run per topology; each simulation repeated 10 times
Measurements	Blocked connections with and without re-optimization Objective function gain Freed OC-units

As can be seen from the previous table, 20 simulations per topology were run. Each run has been repeated 10 times and the average performance metric values have been calculated. We considered several values for the α parameter chosen from the set reported in the previous table and tried out both the breadth and depth search options in local search. Since the main objective of our work is to efficiently address the problem of re-optimization by maximizing the overall network resource usage and hence the medium and long-term carriers' revenues, we were interested in demonstrating the efficiency of the proposed approach on a significant variety of real network topologies through a comparative assessment between our hybrid framework based on Dijkstra SPF for online routing associated to our GRASP-based re-optimization algorithm, and an environment in which no re-optimization was realized. Such assessment focused on their overall effectiveness in term of request rejection ratio/blocking factor, network resource usage optimization and time performances. We did not compare our solution with other re-optimization proposals available in literature due to the peculiarity of our hybrid approach and hence to the lack of comparable results obtained on the same network topologies and traffic distributions. Accordingly, we studied the re-optimization benefits for varying traffic demands and different network layouts. We tried out different static, predefined, or randomly generated traffic demand matrices on several network topologies, both randomly generated and well-known, such as NSFNET and GEANT2 (see Fig. 5) with the bandwidths for the links ranging from OC-1 to OC-768 bandwidth units.

In our tests, each connection request was characterized by a bandwidth demand ranging from OC-1 to OC-12 (622 Mbps) units. We routed these connections using SPF routing. As the network load grows, we continuously monitor the network efficiency expressed by the rejection ratio/blocking factor. When the connections demand exceeds the value of a fixed load threshold we invoke the GRASP re-optimization. We then evaluate the re-optimization gain, comparing the network loads that could be sustained with or without re-optimization. We measured this gain, at varying optimization thresholds, in terms of several quantities. We computed the overall bandwidth gains as the difference (in OC-units) between the total bandwidth available on the network before and after re-optimization and we analyzed the objective function behavior. We also observed the gains in terms of difference between the maximum number of additional end-to-end connections that could be accommodated in the network with and without performing re-optimization. For space limitations and results consistency, we show only the most remarkable results obtained with the well-known NSFNET and GEANT2 networks. In all the presented charts, to make the results more readable and better highlight the evolution trends and properties of the observed performance metrics, the plotted curves have been obtained through polynomial interpolation on the sample observations taken before the beginning and at the end of each GRASP re-optimization step. For the first set of simulations we generated a random demand matrix from all the available source-sink pairs. Next, the network has been loaded by adding end-to-end connections, whose arrival rate is proportional to the values reported for the corresponding pairs in the demand matrix. In Figs. 6 and 7 we show the results in terms of request rejection rate and number of end-to-end connections gain obtained with the first set of simulations on NSFNET in which we used the breadth local search and varying values of $\alpha = \{0.5, 0.2, 0.8\}$ respectively for tests T1, T2 and T3. The simulation without re-optimization is simply indicated with SPF in the figure.

Here, the number of rejected/accepted requests (Y axis) is reported against the number of total generated connection requests (X axis). By looking at the variations in rejection rate as the network was loaded with an increasing number of connections, we can note that, without re-optimization, the network starts rejecting connections much earlier (while the re-optimization approach starts rejecting at about 400 connections) and a substantial and slightly increasing gain can be constantly observed throughout

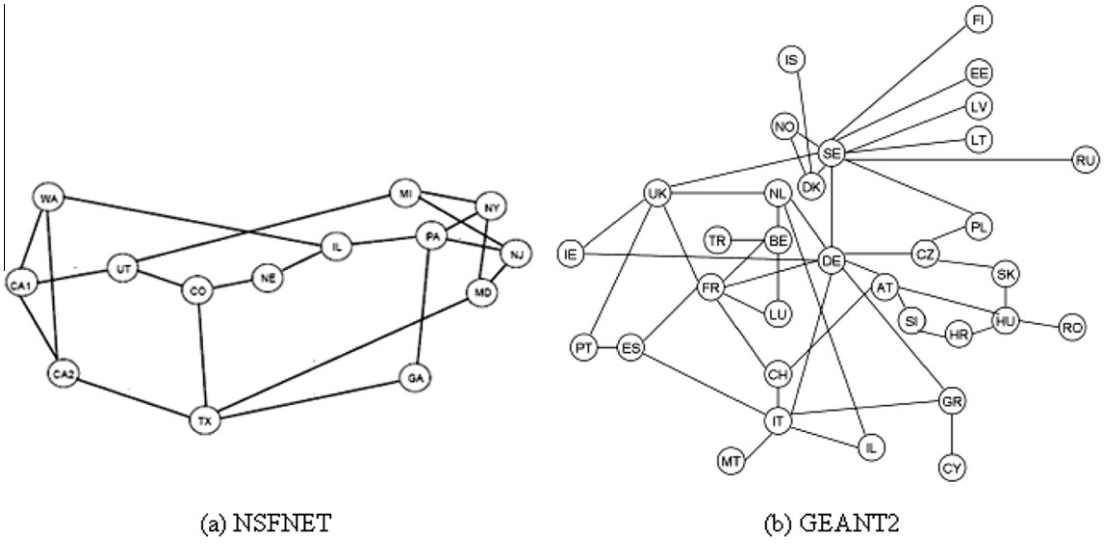

Fig. 5. Sample network topologies used in simulation.

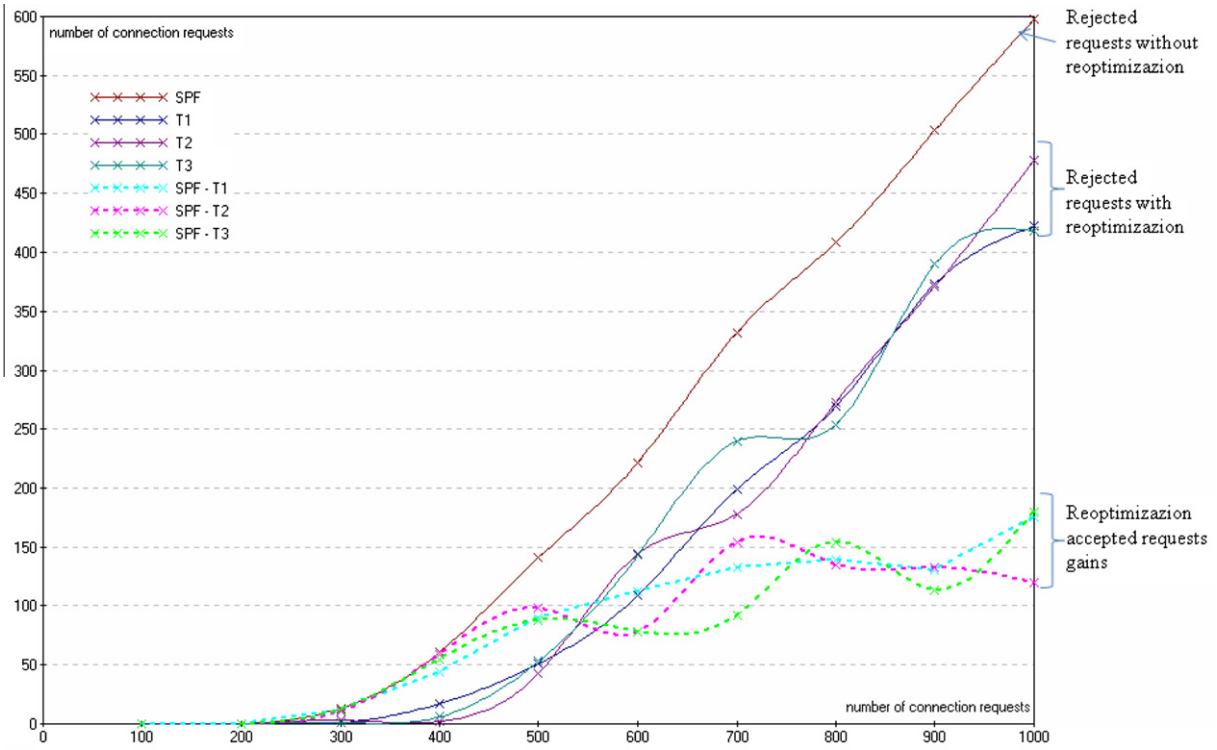

Fig. 6. NSFNET simulation results: blocked connections and re-optimization accepted requests gains.

the experiment also when the load and hence rejection rate increase toward the total network saturation.

Fig. 7 shows the variation in bandwidth gain with increasing network load. The gain starts with a rapid growth and diminishes (dramatically for some α values) after around 50% of the load region; it then restarts increasing and then it progressively decreases, once a local maximum around the 75% of the load is reached, according to an alternate/elastic behavior very common in traffic-related phenomena [40]. The same swinging behavior can also be observed, even if much smoothed, in the above Fig. 6

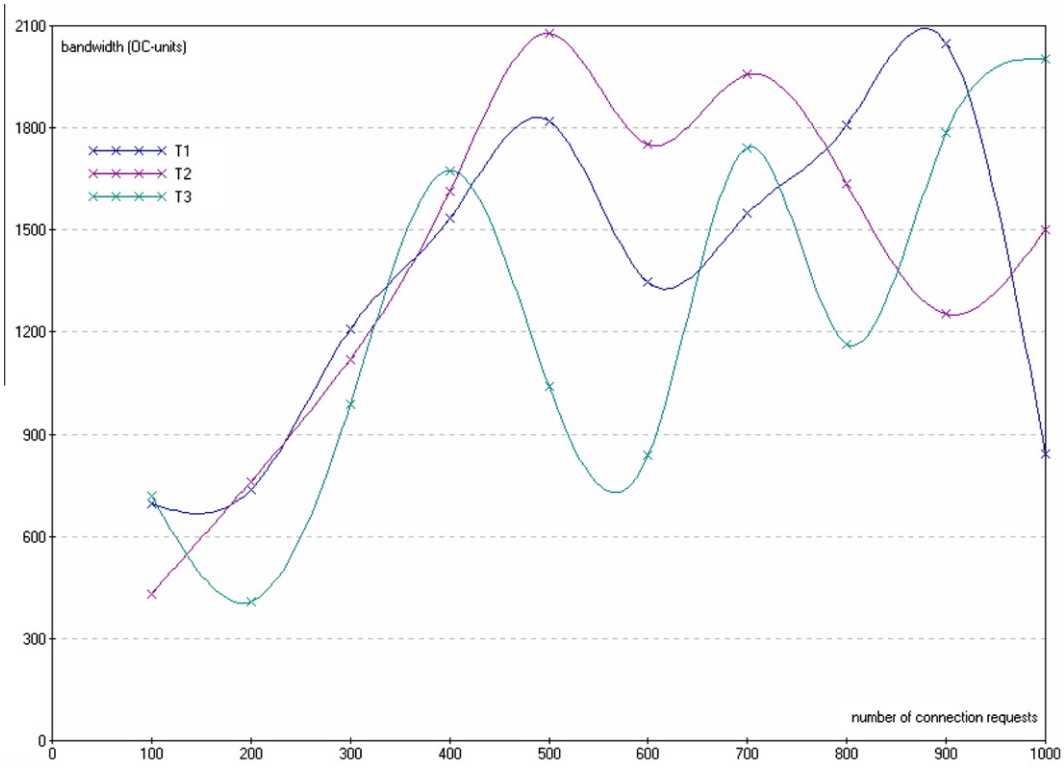

Fig. 7. NSFNET simulation results: bandwidth gains in OC-units.

connection gain curves. The reason for this can be attributed to the reduced flexibility in moving existing paths to other routes because of the periodical reduction in links' spare capacities. This observation suggests that, in a dynamic lightpath routing scenario, the re-optimization procedure should be invoked before the network load reaches 50% (or some other value, dependent on the network specific characteristics). Waiting for the first request to be blocked before attempting re-routing might be too late and reduce the overall efficiency of the re-optimization process. The second set of simulations experiments analyzes the sensitivity of the re-optimization scheme with respect to variations in network topology (and hence link bandwidth) for a fixed demand matrix. Here, the connection requests are distributed on all the network nodes according to the probability distribution obtained from the traffic matrices given in [41] for NSFNET and in [42] for GEANT2, where traffic volumes have been scaled proportionally to the traffic distribution. In Figs. 8 and 9 we show the results obtained with the second set of simulations on GEANT2 in which we used the depth local search and varying values of $\alpha = \{0.5, 0.2, 0.8\}$ respectively for tests T4, T5 and T6. The simulation without re-optimization is simply indicated with SPF in the figure.

When observing the variations in both connections and available bandwidth gain as the network topology changes towards a more connected mesh with higher capacity links (for this sake we can compare the NSFNET and GEANT2 behaviors respectively shown in Figs. 6–9) we can evidence a certain progressive decline in the gain increasing with the network size. This effect highlights that re-optimization algorithms are more efficient in finding good solutions in narrower networks with fewer resources available, since networks with a larger number of links and more capacity have in general less potential for offline reconfiguration gain, as even simple online routing schemes can easily produce acceptable solutions under a physiological load. In fact, in a lightly loaded network, with a lot of available links and capacities, the re-optimization effect allows the admission of only a few additional connection requests while, in an overloaded network, where resource become scarce, the number of additional connections that can be routed on a re-optimized logical topology greatly increases. This behavior is due to the fact that when there is plenty of connectivity resources the number of connections rejected by traditional routing algorithms such as SPF greatly increases, so that any re-optimization strategy – assumed to work on the same traffic matrix – has a much larger potential to satisfy the requests that were previously rejected.

We observe from both Figs. 6 and 9 that the best results in terms of bandwidth and connection gains have been obtained by working with a good balance between greediness and randomness (T1, T4 with $\alpha = 0.5$), in which gains follow a more linear trend, whereas an almost greedy or random selection process (respectively T2, T5 with $\alpha = 0.2$ and T3, T6 with $\alpha = 0.8$) exhibits a sinusoidal-like behavior due to the myopic choices done by the use of a too small or too large RCL. Obviously, a linear trend is much preferable with respect to a non-linear one as it achieves better average results in terms of higher bandwidth gains and lower blocking probability. Similar results have been obtained in breadth and depth local searches, showing that local optima may be reached in both procedures thanks to the robust approach of the Greedy meta-heuristic. Finally, in Fig. 10 we plotted the average re-optimization times for the execution of the illustrated tests referred to the parallel implementation of the Grasp procedure. The results have been obtained by running one search thread on each network node and measuring the starting and ending times. Resulting times have been averaged by the number of nodes and reported in Fig. 10 against the number of connection requests. Communication times among nodes have not been considered in the measurement. As we can see, even in GEANT2, which is a more complex network than NSFNET, the computational times are all below the 1 s threshold which is an affordable delay time for a network [2]. Consequently, the proposed re-optimization strategy is suitable to be implemented on a

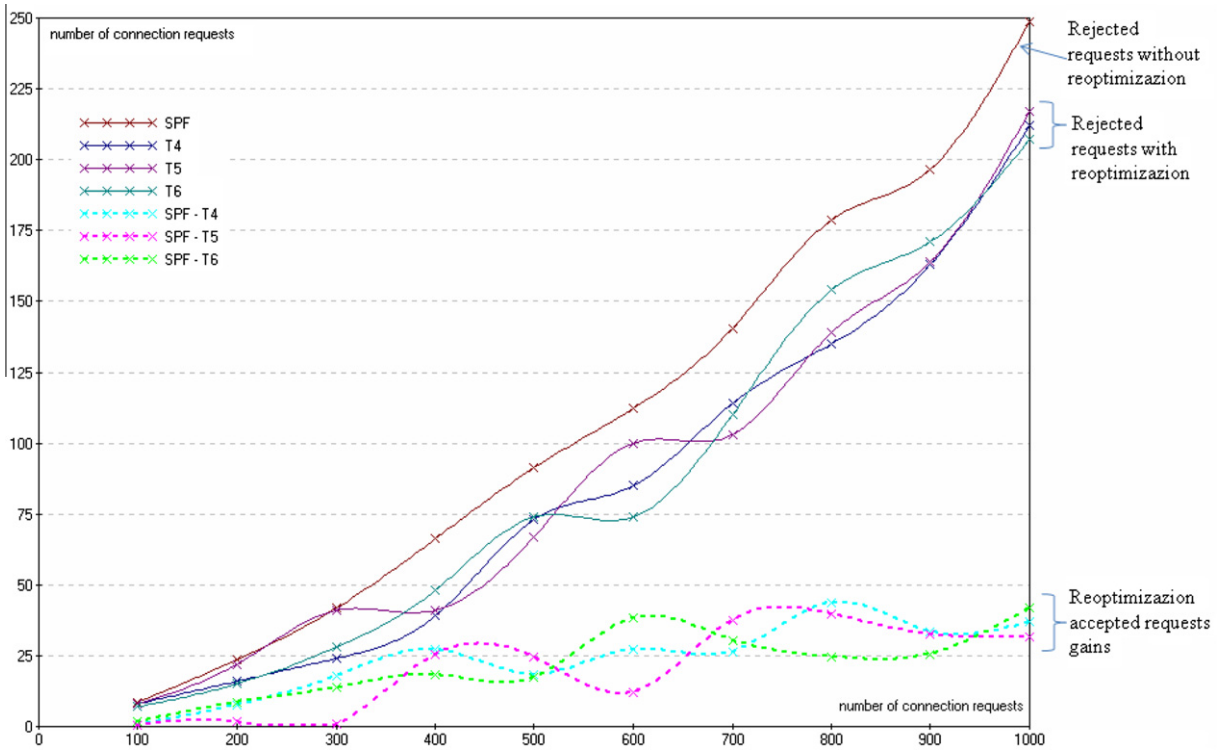

Fig. 8. GEANT2 simulation results: blocked connections and re-optimization accepted requests gains.

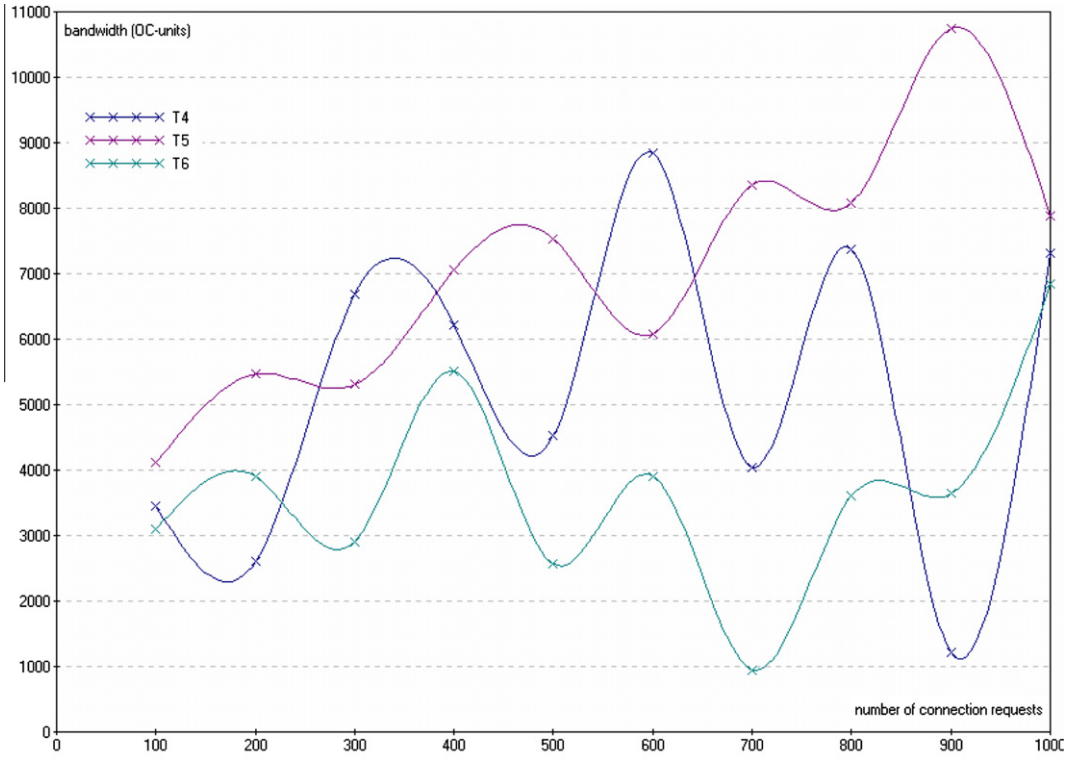

Fig. 9. GEANT2 simulation results: bandwidth gains in OC-units.

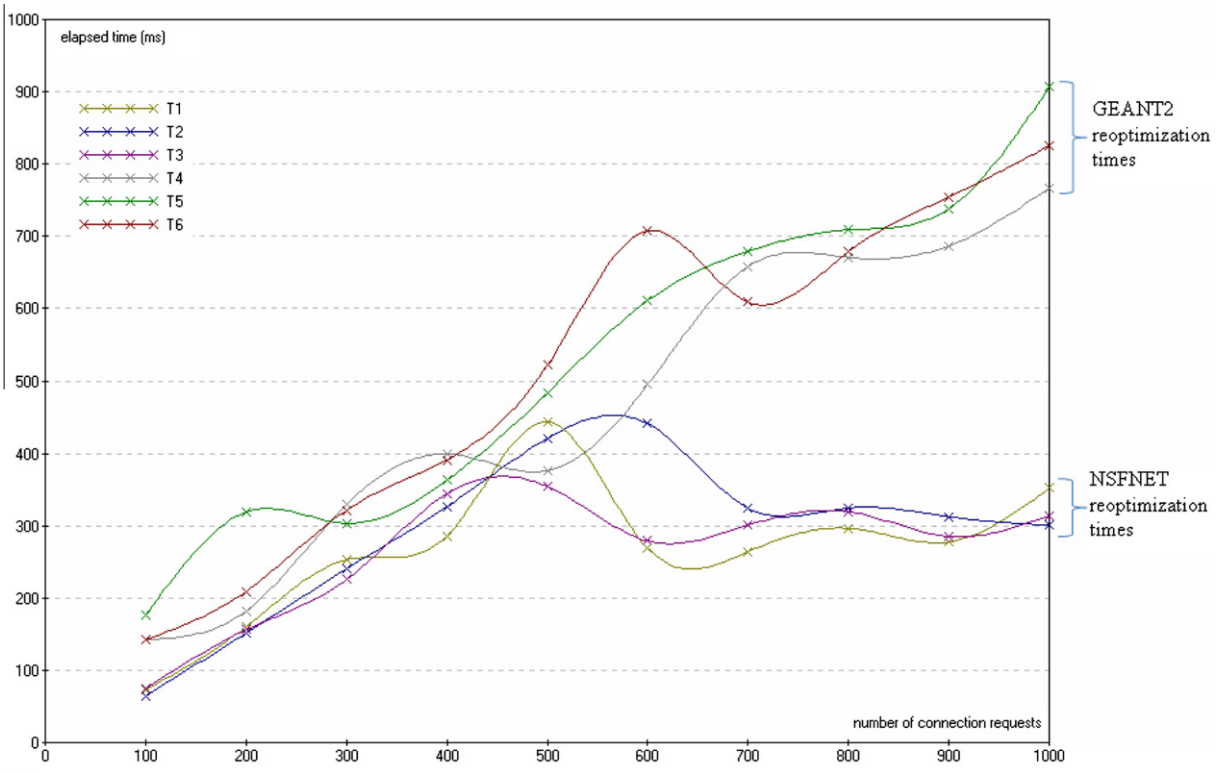

Fig. 10. GEANT2 and NSFNET simulation results: re-optimization times.

frequent basis, also in a production network, before the network gets totally saturated, so that an acceptable degree of efficiency and service continuity is ensured until it is possible, also at higher loads.

6. Conclusions

Dynamic demands and topology changes caused by the addition/deletion of new links and/or capacity, together with online

routing decisions made on a best-of-now basis, without knowledge of the lightpaths to be set-up in the future, cause the wavelength routing logic to behave sub-optimally, thereby creating opportunities for improvements in network bandwidth efficiency. Lightpath topology re-optimization seizes on these opportunities and offers network operators the ability to better adapt to the network and user requests dynamics. This is achieved by regularly (or upon a particular event) re-routing the existing demands, temporarily eliminating the drift between the current solution and the optimal one that is achievable under the same conditions. Starting from the above premises, we formulated a hybrid approach for integrated online routing and offline reconfiguration of optical networks with sub-wavelength traffic. The key feature of such a scheme is the ability to maintain the network balanced through adaptive on-demand re-optimization by ensuring that a sufficient capacity is kept available between any ingress–egress pair so that the maximum number of connections arriving to the network can be satisfied. The overall focus of this work has been the balancing between the reconfiguration cost (in terms of disturbance to the users' connections already deployed over the network) and a good and simple RWA and grooming solution. We defined a set suitable goals and strategies for an integrated approach, and provided a formulation of the re-optimization procedure based on an iterative refinement process of multiple local search steps structured as a GRASP meta-heuristic procedure. We also developed a heuristic strategy that attempts to achieve minimal disturbance reconfiguration by performing local reconfiguration and delaying as possible the need for global reconfiguration. Furthermore, re-optimization would only occur when needed (when the rejection ratio become unacceptable and the potential savings from re-optimization exceeds some threshold) or upon certain events such as when new links are added or torn down. Simulation results show the notable margins of re-optimization achievable with our approach as well as the time complexity feasibility in real networks such as NSFNET and GEANT2. Rejection ratios of connection set-up requests decreased, allowing more connections to be successfully routed, and bandwidth gains have been observed in all the simulation runs. Besides, we proposed an efficient parallel implementation of GRASP with path-relinking that showed quite linear speedups in the number of processors and such a strategy has been successfully applied to greatly accelerate the proposed re-optimization scheme. In conclusion, the proposed re-optimization schema achieves prominent improvements in network efficiency, with the consequent cost savings.

References

[1] J. Zhou, X. Yuan, A study of dynamic routing and wavelength assignment with imprecise network state information, in: Proceedings of ICPP Workshop on Optical Networks, 2002.

[2] G. Mohan, C. Siva Ram Murthy, A time optimal wavelength rerouting algorithm for dynamic traffic in WDM networks, Journal of Lightwave Technology 17 (3) (1999).

[3] K. Kar, M. Kodialam, T. Lakshman, Minimum interference routing of bandwidth guaranteed tunnels with MPLS traffic engineering applications, IEEE Journal on Selected Areas in Communications 18 (2000).

[4] G.N. Rouskas, M. Ammar, Dynamic reconfiguration in multihop WDM networks, Journal of High Speed Networks 4 (3) (1995) 221–238.

[5] T.A. Feo, M.G.C. Resende, A probabilistic heuristic for a computationally difficult set covering problem, Operations Research Letters 8 (1989) 67–71.

[6] T.A. Feo, M.G.C. Resende, Greedy randomized adaptive search procedures, Journal of Global Optimization 6 (1995) 109–133.

[7] F. Glover, Tabu search and adaptive memory programming—advances, applications and challenges, in: R.S. Barr, R.V. Helgason, J.L. Kennington (Eds.), Interfaces in Computer Science and Operations Research, Kluwer, Boston, 1996, pp. 1–24.

[8] F. Glover, M. Laguna, Tabu Search, Kluwer Academic Publishers, Boston, 1997.

[9] H. Zang, J. Jue, B. Mukherjee, A review of routing and wavelength assignment approaches for wavelength-routed optical wdm networks, Optical Networks Magazine 1 (2000) 47–60.

[10] R. Ramaswami, K.N. Sivarajan, Routing and wavelength assignment in all-optical networks, IEEE/ACM Transactions on Networking 3 (5) (1995) 489–500.

[11] I. Chlamtac, A. Ganz, G. Karmi, An approach to high-bandwidth optical wans, IEEE Transactions on Communications 40 (7) (1992) 1171–1182.

[12] H. Zang, J. Jue, L. Sahasrabuddhe, R. Ramamurthy, B. Mukherjee, Dynamic lightpath establishment in wavelength routed networks, IEEE Communications Magazine 39 (9) (2001) 100–108.

[13] B. Ramamurthy, B. Mukherjee, Wavelength conversion in WDM networking, IEEE Journal Selected Areas in Communications 16 (7) (1998) 1061–1073.

[14] F. Palmieri, U. Fiore, Dynamic network optimization for effective QoS support in large grid infrastructures, in: Lizhe Wang, Jinjun Chen, Wei Jie (Eds.), Quantitative Quality of Service for Grid Computing: Applications for Heterogeneity, Large-scale Distribution, and Dynamic Environments, IGI Global, 2009, pp. 28–48, ISBN: 978-1-60566-370-8.

[15] S. Kirkpatrick, C.D. Gelatt Jr., M.P. Vecchi, Optimization by simulated annealing, Science 220 (4598) (1983) 671–680.

[16] J.H. Holland, Adaptation in Natural and Artificial Systems, MIT Press, Cambridge, MA, USA, 1975.

[17] M.G.C. Resende, C.C. Ribeiro, Greedy randomized adaptive search procedures, in: F. Glover, G. Kochenberger (Eds.), State-of-the-Art Handbook of Metaheuristics, Kluwer Academic Publishers, 2002, pp. 219–249.

[18] F. Glover, M. Laguna, Tabu Search, Kluwer Academic Publishers, Boston, 1997.

[19] F. Glover, Scatter search and path relinking, in: D. Corne, M. Dorigo, F. Glover (Eds.), New Ideas in Optimisation, Wiley, 1999.

[20] F. Glover, M. Laguna, R. Marti, Fundamentals of scatter search and path relinking, Control and Cybernetics 39 (2000) 653–684.

[21] M. Laguna, R. Marti, GRASP and path relinking for the 2-layer straight line crossing minimization, INFORMS Journal on Computing 11 (1999) 44–52.

[22] M.G.C. Resende, C.C. Ribeiro, GRASP with path-relinking for private virtual circuit routing, Networks 41 (2003) 104–114.

[23] C.C. Ribeiro, E. Uchoa, R.F. Werneck, A hybrid GRASP with perturbations for the Steiner problem in graphs, INFORMS Journal on Computing 14 (2002) 228–246.

[24] R.M. Aiex, M.G.C. Resende, P.M. Pardalos, G. Toraldo, GRASP with path relinking for the three index assignment problem, INFORMS Journal on Computing (2003).

[25] M.G.C. Resende, R.F. Werneck, A GRASP with path-relinking for the p-median problem, Technical Report, AT&T Labs Research, Florham Park, NJ, 2002.

[26] K.C. Lee, V.O.K. Li, A circuit rerouting algorithm for all-optical wide area networks, in: Proceedings of IEEE INFOCOM 1994, 1994, pp. 954–961.

[27] D. Banerjee, B. Mukherjee, Wavelength-routed optical networks: linear formulation, resource budgeting tradeoffs, and a reconfiguration study, IEEE/ACM Transactions on Networking 8 (5) (2000) 598–607.

[28] J.F.P. Labourdette, G.W. Hart, A.S. Acampora, Branch-exchange sequences for reconfiguration of lightwave networks, IEEE Transactions on Communications 42 (10) (1994) 2822–2832.

[29] I. Baldine, G.N. Rouskas, Dynamic reconfiguration policies for WDM networks, in: Proceedings of IEEE INFOCOM 1999, 1999.

[30] M. Sridharan, A.K. Somani, M.V. Salapaka, Approaches for capacity and revenue optimization in survivable WDM networks, Journal of High Speed Networks 10 (2) (2001) 109–125.

[31] E. Bouillet, J.F. Labourdette, R. Ramamurthy, S. Chaudhuru, Lightpath re-optimization in mesh optical networks, IEEE/ACM Transactions on Networking 13 (2) (2005) 437–447.

[32] R. Bhatia, M.S. Kodialam, T.V. Lakshman, Fast network reoptimization schemes for mpls and optical networks, Computer Networks 50 (3) (2006) 317–331.

[33] J.Y. Zhang, O.W.W. Yang, J. Wu, M. Savoie, Optimization of semi-dynamic lightpath rearrangements in a WDM network, IEEE Journal on Selected Areas in Communications 25 (9) (2007) 3–17.

[34] D.R. Din, Solving virtual topology reconfiguration problem on survivable WDM networks by using simulated annealing and genetic algorithms, Photonic Network Communications 18 (1) (2009) 1–13.

[35] Y. Xin, M. Shayman, R.J. La, S.I. Marcus, OPNp1-2. Reconfiguration of survivable MPLS/WDM networks, in: Proceedings of IEEE GLOBECOM 2006, 2006.

[36] M. Kodialam, T.V. Lakshman, Minimum interference routing with applications to MPLS traffic engineering, in: Proceedings of IEEE Infocom, 2000.

[37] M.G.C. Resende, C.C. Ribeiro, GRASP and path-relinking: recent advances and applications, in: T. Ibaraki, K. Nonobe (Eds.), Metaheuristics: Progress as Real Problem Solvers, Springer, 2005.

[38] D.S. Johnson, C.H. Papadimitriou, M. Yannakakis, How easy is local search?, Journal of Computer and System Sciences 17 (1) (1998) 79–100

[39] F. Palmieri, U. Fiore, S. Ricciardi, SimulNet: a wavelength-routed optical network simulation framework, in: Proceedings of IEEE ISCC 2009, 2009, pp. 281–286.

[40] C. Casetti, R. Lo Cigno, M. Mellia, M. Munafò, Z. Zsóka, A realistic model to evaluate routing algorithms in the internet, in: Proceedings of IEEE Globecom 2001, 2001, pp. 1882–1885.

[41] R. Ramaswami, K.N. Sivarajan, Design of logical topologies for wavelength-routed optical networks, IEEE Journal on Selected Areas in Communications 14 (1996) 840–851.

[42] S. Uhlig, B. Quoitin, J. Lepropre, S. Balon, Providing public intradomain traffic matrices to the research community, ACM SIGCOMM Computer Communication Review 36 (1) (2006) 83–86.